Katherine Vidal (SBN 194971 / vidal@fr.com)
Betty H. Chen (SBN 290588 / bchen@fr.com)
Matthew R. McCullough (SBN 301330 / mccullough@fr.com)
FISH & RICHARDSON P.C.
500 Arguello Street, Suite 500
Redwood City, CA 94063
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

John P. Brinkmann (brinkmann@fr.com)
1221 McKinney Street, Suite 2800
Houston, TX 77010
Telephone: (713) 654-5300
Facsimile: (713) 652-0109

Attorneys for Defendant
APPLE INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

(SAN FRANCISCO DIVISION)

REALTIME DATA, LLC D/B/A/ IXO, Plaintiff, v. APPLE INC., Defendant(s).	Case No. 4:16-cv-02595-JD **DECLARATION OF CRAIG WILLS**

I, Craig E. Wills, declare as follows:

1. I have extensive experience in computers and computer operating systems spanning a career of over thirty years in the field. My most relevant expertise includes operating systems, distributed systems, networking and user interfaces. I have been a professor of Computer Science for over 25 years. I am currently the Department Head of the Computer Science Department at the Worcester Polytechnic Institute. I also have industry experience working as an engineer at AT&T Bell Laboratories.

2. I earned my B.S. in Computer Science from the University of Nebraska in 1982, and my M.S. and Ph.D. in Computer Science from Purdue University in 1984 and 1988, respectively

3. The details of my education and professional activities are attached to this Declaration as Attachment A.

4. I have been asked to provide this declaration in support of Apple's Opening Claim Construction Brief in Case No. 4:16-cv-02595-JD. I have reviewed Realtime's asserted patents: U.S. Patent Nos. 7,181,608 ("the '608 patent"), 8,090,936, and 8,880,862, their prosecution histories, Provisional Application 60/180,114, and Realtime's Opening Claim Construction Brief.

5. I understand that claim construction, and determining how a person of ordinary skill in the art would understand the claim terms in view of the intrinsic evidence is a legal question that the Court alone decides. I have been informed by counsel that Realtime's asserted patents have a priority date of February 3, 2000. I believe one of ordinary skill in the art of computer storage systems around February 3, 2000 would have had at least a bachelor's degree in computer science or engineering and 1-2 years of professional experience in computer system design, or equivalent educational and professional experience.

6. Therefore, my analysis as set forth below, is offered to respond to Realtime's opening claim construction brief, where I believe a person of ordinary skill in the art would differ as to the technical assertions, to explain the understanding of a person of ordinary skill in the art of certain terms that are well-known terms of art, and to explain the import of the technical disclosure of the patents and their provisional application.

7. I understand that Realtime alleges that its patents "claim inventions for decreasing the boot time of a computer." Dkt. 73 at 2.

8. I understand that Realtime alleges that a problem existed because "[t]raditionally, computer boot-up was performed by instructing the computer to retrieve and load the operating system from a hard disk. But, these hard disks were and generally are slowest component of the computer, resulting in delays and long start-up times." *Id.* at 3 (citing '608 patent at 1:43-52 ("The data storage and retrieval bandwidth of mass storage devices, however, is typically much less as

compared with the bandwidth of other elements of a computing system. Indeed, over the last decade, although computer processor performance has improved by at least a factor of 50, magnetic disk storage performance has only improved by a factor of 5. Consequently, memory storage devices severely limit the performance of consumer, entertainment, office, workstation, servers, and mainframe computers for all disk and memory intensive operations."); 2:4-14 ("Magnetic disk mass storage devices currently employed in a variety of home, business, and scientific computing applications suffer from significant seek-time access delays along with profound read/write data rate limitations. Currently the fastest available disk drives support only a sustained output data rate in the tens of megabytes per second data rate (MB/sec). This is in stark contrast to the modern Personal Computer's Peripheral Component Interconnect (PCI) Bus's low end 32 bit/33 Mhz input/output capability of 264 MB/sec and the PC's internal local bus capability of 800 MB/sec.") 2:15-37 ("Another problem within the current art is that emergent high performance disk interface standards such as the Small Computer Systems Interface (SCSI-3), Fibre Channel, AT Attachment UltraDMA/66/100, Serial Storage Architecture, and Universal Serial Bus offer only higher data transfer rates through intermediate data buffering in random access memory. These interconnect strategies do not address the fundamental problem that all modern magnetic disk storage devices for the personal computer marketplace are still limited by the same typical physical media restrictions. In practice, faster disk access data rates are only achieved by the high cost solution of simultaneously accessing multiple disk drives with a technique known within the art as data striping and redundant array of independent disks (RAID). RAID systems often afford the user the benefit of increased data bandwidth for data storage and retrieval. By simultaneously accessing two or more disk drives, data bandwidth may be increased at a maximum rate that is linear and directly proportional to the number of disks employed. Thus another problem with modern data storage systems utilizing RAID systems is that a linear increase in data bandwidth requires a proportional number of added disk storage devices"); 21:40-44 ("Since most boot devices are relatively slow compared to the speed of most computer busses, a long delay is seen by the computer user.")).

9. I understand that Realtime alleges that it solved this problem by using "compression and decompression" such that "[a] resultant speed increase is achieved, in large measure, because compressed data requires less space on the hard drive than uncompressed data. Less data to read from the hard drive allows the operating system (or application) to load into the computer's memory faster." *Id.* (citing '608 patent at 8:57-9:3 ("It is to be appreciated that the implementation of a storage controller according to the present invention significantly accelerates the performance of a computer system and significantly increases hard disk data storage capacity. For instance, depending on the compression rate, for personal computers running standard Microsoft Windows® based business application software, the storage controller provides: (1) an increase of n:1 in disk storage capacity (for example, assuming a compression ratio of 3:1, a 20 gigabyte hard drive effectively becomes a 60 gigabyte hard drive) (2) a significant decrease in the computer boot-up time (turn-on and operating system load) and the time for loading application software and (3) User data storage and retrieval is increased by a factor of n:1."); 19:10-20:9 ("The onboard cache of the data storage controller is shared by the DSP, Disk Interface, and PCI Bus. The best case, maximum bandwidth for the SDRAM memory is 70 megawords per second, or equivalently, 280 megabytes per second. The 32 bit PCI Bus interface has a best case bandwidth of 132 megabytes per second, or equivalently 33 megawords per second. In current practice, this bandwidth is only achieved in short bursts. The granularity of PCI data bursts to/from the data storage controller is governed by the PCI Bus interface data buffer depth of sixteen words (64 bytes). The time division multiplexing nature of the current PCI Data Transfer Buffering methodology cuts the sustained PCI bandwidth down to 66 megabytes/second. Data is transferred across the ultraDMA disk interface at a maximum burst rate of 66 megabytes/second. It should be noted that the burst rate is only achieved with disks that contain onboard cache memory. Currently this is becoming more and more popular within the industry. However assuming a disk cache miss, the maximum transfer rates from current disk drives is approximately six megabytes per second. Allotting for technology improvements over time, the data storage controller has been designed for a maximum sustained disk data rate of 20 megabytes second (5 megawords/second). A design challenge is created by the need for continuous access to

the SDRAM memory. Disks are physical devices and it is necessary to continuously read data from disk and place it into memory, otherwise the disk will incur a fall rotational latency prior to continuing the read transaction. The maximum SDRAM access latency that can be incurred is the depth of the each of the two disk FIFO s or sixteen data. Assuming the FIFO is sixteen words deep the maximum latency time for emptying the other disk FIFO and restoring it to the disk interface is sixteen words at 5 megawords per second or (16×3.2 usec)=1 usec. Each EMIF clock cycle is 14.2857 nsec, thus the maximum latency translates to 224 clock cycles. It should be noted that transfers across the disk interface are 16 bits wide, thus the FPGA is required to translate 32 bit memory transfers to 16 bit disk transfers, and vice-versa. The DSP services request for its external bus from two requestors, the Enhanced Direct Memory Access (EDMA) Controller and an external shared memory device controller. The DSP can typically utilize the full 280 megabytes of bus bandwidth on an 8 k through 64 K byte (2 k word through 16 k word) burst basis. It should be noted that the SDRAM memory is not utilized for interim processing storage, and as such bandwidth is only utilized in direct proportion to disk read and write commands. For a single read from disk transaction data is transferred from and DMA transfer into SDRAM memory. This data is then DMA transferred by the DSP into onboard DSP memory, processed, and re transferred back to SDRAM in decompressed format (3 words for every one word in). Finally the data is read from SDRAM by the PCI Bus Controller and placed into host computer memory. This equates to eight SDRAM accesses, one write from disk, one read by the DSP, three writes by the DSP and three by the PCI Bus. Disk write transactions similarly require eight SDRAM accesses, three from the PCI, three DSP reads, one DSP write, and one to the disk. Neglecting overhead for setting up DMA transfers, arbitration latencies, and memory wait states for setting up SDRAM transactions, the maximum DSRA theoretical SDRAM bandwidth limit for disk reads or writes is 280/8 megabytes second or 35 megabytes second.”))

10. However, the problem purportedly solved by Realtime’s claimed data storage technology is far less of an issue today. First and foremost, most computing devices made in 2007 and onwards (“modern computing devices”) are typically no longer “shut down” when the device is

not in use, but rather are simply put into sleep (or hibernate) mode. Therefore, when the device is put into use again, it is simply "awoken" from sleep mode rather than "booted up." Sleep mode maintains the data currently in a computer's memory, so it does not require the transfer of operating system data from a hard disk, and therefore the "bottleneck" problem targeted by Realtime's patents is avoided entirely.

11. Similarly, modern computing devices no longer need to be restarted (or booted) as frequently as they did in the late 1990s and early 2000s. One reason for this is that current operating systems are far more robust and reliable than those in the past. Early operating systems would need to be restarted as often as multiple times daily in order to resolve issues such as poor performance or the system hanging. Modern operating systems typically need not be restarted more than a few times a month (or even less). Therefore, modern computing device do not need to be restarted – and therefore encounter the boot up process – nearly as often as computing devices in the late 1990s and early 2000s did.

12. Another factor that previously contributed to a higher number of device boot ups was mobile computing device short battery life. When devices had a short battery life, there was a higher chance that the battery would run out and that the device would power off. Once such a device was powered-up again, it would have to be booted up again before it could be used. Battery life and power consumption of mobile computing devices such as a laptops and mobile phones has improved greatly since the late 1990s and early 2000s. Thus modern mobile devices do not need to boot up as often since they do not run out of battery as frequently.

13. Moreover, even in the relatively rare instances when a modern computing device needs to be "booted up," advances in data transfer technologies have largely eliminated any practical limiting effect that a data transfer rate "bottleneck" could have on boot up time. Both hard disk data transfer speeds and bus transfer speeds have increased dramatically. For example, hard disk drives have been replaced with SSDs, or Solid State Drives, a technology that does not suffer from the same latency and speed limitations as legacy hard drives. For example, rather than a hard disk not taking advantage of the full speed of the hard disk interface (as in Realtime's example), a modern

SSD can even be limited to the available speed of the hard disk interface it is using. A modern SSD can be thirty to ninety times faster than the hard disks referenced in the Realtime patents. Even more recent storage technology, such as NVMe, bypasses the legacy hard disk controller entirely and is able to operate at speeds matching a more modern peripheral bus such as PCI Express, which is nearly a hundred and forty times faster than the example "fastest available" hard disk mentioned in Realtime's patents. Moreover, main memory (aka random access memory or "RAM") data transfer rates and CPU processing power has likewise greatly increased during this period. Indeed, all system components that could affect boot time have gotten significantly faster and, unsurprisingly, boot times have also gotten faster. Therefore, even in the rare instances that a modern computing device needs to be booted up, there is not a pressing need to resolve a data transfer bottleneck in attempt to increase boot speed.

14. I understand that Realtime has proposed that the terms "initialization" / "initializing" / "prior to completion of initialization of the central processing unit" / "after completion of initialization of the central processing unit" either not be construed or construed as "prepare for use." Dkt. 73 at 8.

15. A CPU can go through many states between, for example, power-up of the CPU and CPU execution of operating system code. Such states may include: power-up, power stabilization, self-check, enabling CPU registers, enabling CPU cache, configuring CPU cache, executing software initialization code, enabling virtual memory system, enabling memory array, executing operating system code. Depending on what the CPU is needed for, one of ordinary skill in the art could understand the CPU and/or the computer system to be "initialized" upon reaching any of these states.

16. Similarly, one of ordinary skill in the art could consider the CPU to be "prepared for use," upon reaching any of these states, again depending on what "use" the processor is being put to. For example, a CPU could be "prepared for use" as soon as the CPU is ready to take any action, for example executing a built in self-test ("BIST"). [*See* APL-REAL_00556387-88]. Or a CPU could be "prepared for use" when it is ready to execute code. However, there are different types of

code that can be executed by a CPU, including (for example) software initialization code and operating system code. Alternatively, a CPU could be "ready for use" when it is ready to execute code in response to user input. For similar reasons, one of ordinary skill in the art could have the same understanding regarding a computer system being prepared for use.

17. Moreover, a person of ordinary skill in the art would understand a CPU to be "initialized" or the "completion of initialization of the central processing unit" to have taken place depending on several factors, including but not limited to: (1) the type of CPU being initialized; (2) the purpose that the CPU serves; (3) whether there is a single CPU or multiple CPUs; (4) whether a hard reset or soft reset is performed on the CPU.

18. Whether a CPU has been initialized depends on the type of CPU in question. For example, Intel IA-32 architecture processors all went through the following sequence after power-up or assertion of the RESET# pin: "each processor on the system bus performs a hardware *initialization* of the processor (known as a hardware reset) and an optional built-in self-test (BIST). A hardware reset sets each processor's registers to a known state and places the processor in real-address mode. It also invalidates the internal caches, translation lookaside buffers (TLBs) and the branch target buffer (BTB)." [APL-REAL_00556387]. **The next step in the sequence depended on which processor family the CPU was a part of**. *See id*. ("At this point, the action taken depends on the processor family.) For example, Pentium 4 processors executed a "multiple processor (MP) initialization process, which selects one processor as a "bootstrap processor," which then immediately starts to execute software initialization code. *Id.* In contrast, earlier generation Pentium Processors pre-designated one processor as the primary processor which, immediately began to execute software initialization code. *Id.* Therefore, depending on whether the CPU in question was a Pentium 4 processor or an earlier generation Pentium processor, one of ordinary skill in the art would understand that CPU to have been initialized depending on whether that CPU had undergone the "multiple processor initialization process."

19. The purpose that a CPU serves also influences whether a person of ordinary skill in the art would understand the CPU to be "initialized." For example, the Intel processors described

above can go through "an optional built-in self-test (BIST)." [APL-REAL_00556387-88]. Whether this optional part of the initialization process takes place could depend on the computing environment that the CPU is placed in. For example, a CPU could be located in a consumer environment where speed of initialization is the primary concern, such as a laptop. In such a context, the CPU could be configured not to execute a BIST upon power-on. In other contexts, such as a mission-critical server environment, fault detection might be prioritized over the speed of initialization and the CPU would be required to execute a BIST every time it is powered-on. Therefore, depending on the purpose and environment of the CPU, one of ordinary skill in the art may or may not understand the CPU to have been initialized depending on whether that CPU has executed a BIST, for example.

20. Whether a CPU has initialized also depends on whether a single or multiple processors are being initialized. For example, certain Pentium processors employ a multiple processor initialization protocol ("MP"). [*See* APL-REAL_00556380; APL-REAL_00556387]. This protocol defines two classes of processors: a bootstrap processor ("BSP") and application processors ("AP"). [*See* APL-REAL_00556380]. A bootstrap processor is dynamically from among the multiple processor by an MP initialization algorithm. *Id.* The bootstrap processor configures the APIC environment, and starts the secondary processors, under software control. *Id.* In contrast the AP processor(s) complete a minimal self-configuration, then wait for a startup signal from the BSP processor. *Id.* Upon receiving a startup signal, an AP completes its configuration. *Id.* Based on this, one of ordinary skill in the art's understanding of whether a CPU has been initialized could vary depending on whether that CPU is a "bootstrap processor" or an "application processor."

21. Whether a CPU has initialized might also depend on whether a hard or soft reset has been performed on the CPU. When a hard reset has been performed, for example when a computer is powered up, a CPU may go through all of the steps described in the preceding paragraphs [*See supra* ¶¶ 15, 18]. However, when a soft reset is performed, for example when a computer comes out of "sleep mode," a CPU will go through different and potentially fewer steps. For example,

during a soft reset, the CPU would not test the memory controller as that would wipe the content of the memory and interfere with whatever the user of the computer was doing prior to sleep. As such, depending on whether the CPU is initializing after a hard or soft reset, one of ordinary skill in the art may or may not understand the CPU to have been "initialized" depending on whether a test of the memory controller has been performed.

22. The terms "initialization" / "initializing" / "prior to completion of initialization of the central processing unit" / "after completion of initialization of the central processing unit," have no definite meaning to a person of ordinary skill in the art.

23. I have provided my opinion to the best of my knowledge at this time. I reserve the right to supplement my opinion at a future date.

Date: March 10, 2017

Craig E. Wills, Ph.D.